# PE-6 – Parsing and Formatting

## Objective

Create a simple number guessing game. Your program will generate a random integer number between 0 and 100 (inclusive) and a player will try to guess it.

## Details:

All work for this program will be done in the main method. Your code should do the following:

1. Generate a random integer number in the range of 0 – 100 (inclusive, meaning it could generate 0, 100, or any other integer in between)
   1. For now, use the following code in the beginning of your Main method:

Random rand = new Random();

// generate a random number between 0 inclusive and 101 exclusive

int randomNumber = rand.Next(0, 101);

1. Print this random number at the top of your program.
   1. This will help you test your program for guesses that are too high, too low, or correct.
2. Use a loop to give the user 8 tries to guess a number by doing the following:

* Prompt the user to guess a number.
* Parse the value the user enters. Check if their input is valid.
  + *Valid guesses* are between 0 and 100. *Invalid guesses* are below 0, above 100 or non-integer values.
  + If their guess is invalid, make them re-enter it until they enter a valid number. An invalid guess does not use up one of the user’s guesses. (You may want to use a while loop for this)
* After each guess, tell them if their guess was **correct**, **high** or **low**.
* If they guess the correct number before their 8 turns are up:
  + Tell them how many turns it took and end the loop.

1. If they fail to guess the correct number, tell them what it was.

## Stuck? Break it down into small steps:

Generate a random number. Print to the Console window to test that it’s working.

Create a loop that runs 8 times. Print something each of the 8 times to test that it’s working.

Inside that loop, ask the user for a number between 1 and 100. Test that.

Next, check the user’s input to ensure it’s between 1 and 100. Test that.

Create a loop that will continually ask the user for good data if their data is invalid. Test that.

Check the user’s data against the random number – print a message when the user’s guess is too high, too low, or correct. Test.

Outside of the 8-guess loop, print the number of turns it took the user to guess the correct number. Test.

## Sample Runs:

### Correct run:

37

Turn #1: Enter your guess: 50

Too high

Turn #2: Enter your guess: 25

Too low

Turn #3: Enter your guess: 37

Correct! You won in 3 turns.

### Too many turns:

66

Turn #1: Enter your guess: 1

Too low

Turn #2: Enter your guess: 2

Too low

Turn #3: Enter your guess: 3

Too low

Turn #4: Enter your guess: 4

Too low

Turn #5: Enter your guess: 5

Too low

Turn #6: Enter your guess: 6

Too low

Turn #7: Enter your guess: 7

Too low

Turn #8: Enter your guess: 8

Too low

You ran out of turns. The number was 66.

### Bad input:

6

Turn #1: Enter your guess: 1

Too low

Turn #2: Enter your guess: 2

Too low

Turn #3: Enter your guess: 3

Too low

**Turn #4: Enter your guess: -4**

**Invalid guess – try again**

Turn #4: Enter your guess: 4

Too low

Turn #5: Enter your guess: 5

Too low

**Turn #6: Enter your guess: 999**

**Invalid guess – try again**

Turn #6: Enter your guess: 6

Correct! You won in 6 turns.

## Submission:

Make sure you follow the coding standards for all code you create.

Add, Commit and Push your project to your GitHub repository.

Submit your project URL to the appropriate dropbox.

## Optional Challenge:

1. Use your imagination! What can you do to make your guessing game better? ☺